(Conic) Linear Optimization: Problem Instances II

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/~yyye

LY 5th, Chapter 1, Chapter 2.1-2.2 Chapter 6

Prediction Market I: World Cup Information Market

Order:	#1	#2	#3	#4	#5
Argentina	1	0	1	1	0
Brazil	1	0	0	1	1
Italy	1	0	1	1	0
Germany	0	1	0	1	1
France	0	0	1	0	0
Bidding Prize: π	0.75	0.35	0.4	0.95	0.75
Quantity limit: $\mathbf{q}$	10	5	10	10	5
Order fill: $\mathbf{x}$	x_1	x_2	x_3	x_4	x_5

Prediction Market II: Call Auction Mechanism

Given m potential states that are mutually exclusive and exactly one of them will be realized at the maturity.

An order is a bet on one or a combination of states, with a price limit (the maximum price the participant is willing to pay for one unit of the order) and a quantity limit (the maximum number of units or shares the participant is willing to accept).

A contract on an order is a paper agreement so that on maturity it is worth a notional $$1$ dollar if the order includes the winning state and worth $$0$ otherwise.

There are n orders submitted now.

Prediction Market III: Input Order Data

ith order

The ith order is given as $(\mathbf{a}_{i\cdot} \in R^m_+,\ \pi_i \in R_+,\ q_i \in R_+)$: $\mathbf{a}_{i\cdot}$ is the betting indication row vector where each component is either 1 or 0

$$\mathbf{a}_{i\cdot} = (a_{i1},\ a_{i2},\ \ldots,\ a_{im})$$

where 1 is winning state and 0 is non-winning state; π_i is the price limit for one unit of such a contract, and q_i is the maximum number of contract units the better like to buy.

Prediction Market IV: Output Order-Fill Decisions

Let x_i be the number of units or shares awarded to the ith order. Then, the ith bidder will pay the amount $\pi_i \cdot x_i$ and the total amount collected would be $\pi^T \mathbf{x} = \sum_i \pi_i \cdot x_i$.

If the jth state is the winning state, then the auction organizer need to pay the winning bidders

$$\left(\sum_{i=1}^{n} a_{ij} x_i \right) = \mathbf{a}_{.j}^T \mathbf{x}$$

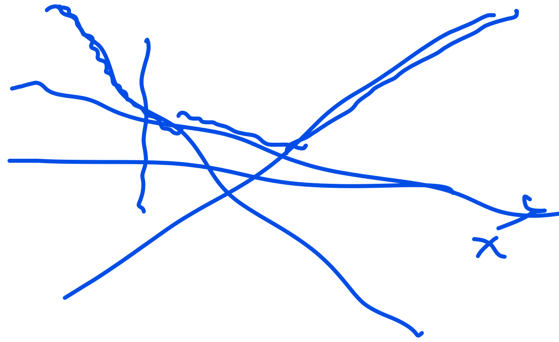
where column vector

$$\mathbf{a}_{.j} = (a_{1j};\ a_{2j};\ \ldots;\ a_{nj})$$

The question is, how to decide $\mathbf{x} \in R^n$, that is, how to fill the orders.

Prediction Market V: Worst-Case Profit Maximization

$$\begin{array}{lrl} \max & \pi^T\mathbf{x} - \max_j\{\mathbf{a}_{\cdot j}^T\mathbf{x}\} & \\ \text{s.t.} & \mathbf{x} & \le \mathbf{q}, \\ & \mathbf{x} & \ge \mathbf{0}. \end{array}$$



$$\begin{array}{lrl} \max & \pi^T\mathbf{x} - \max(A^T\mathbf{x}) & \\ \text{s.t.} & \mathbf{x} & \le \mathbf{q}, \\ & \mathbf{x} & \ge \mathbf{0}. \end{array}$$

This is NOT a linear program.

Prediction Market VI: LP Representation

However, the problem can be rewritten as

$$
\begin{array}{lrl}
\max & \pi^T \mathbf{x} - y & \\
\text{s.t.} & A^T\mathbf{x} - \mathbf{e}\cdot y & \leq \mathbf{0}, \\
 & \mathbf{x} & \leq \mathbf{q}, \\
 & \mathbf{x} & \geq \mathbf{0},
\end{array}
$$

where $\mathbf{e}$ is the vector of all ones. This is a linear program.

$$
\begin{array}{lrll}
\max & \pi^T \mathbf{x} - y & & \\
\text{s.t.} & A^T\mathbf{x} - \mathbf{e}\cdot y + s_0 & = \mathbf{0}, & \\
 & \mathbf{x} + \mathbf{s} & = \mathbf{q}, & \\
 & (\mathbf{x}, s_0, \mathbf{s}) & \geq \mathbf{0}, & y \text{ free},
\end{array}
$$

Max-Cut Problem

This is the Max-Cut problem on an undirected graph $G = (V, E)$ with non-negative weights w_{ij} for each edge in E (and $w_{ij} = 0$ if $(i, j) \notin E$), which is the problem of partitioning the nodes of V into two sets S and $V \setminus S$ so that

$$w(S) := \sum_{i \in S,\, j \in V \setminus S} w_{ij}$$

is maximized. A problem of this type arises from many network planning, circuit design, and scheduling applications.

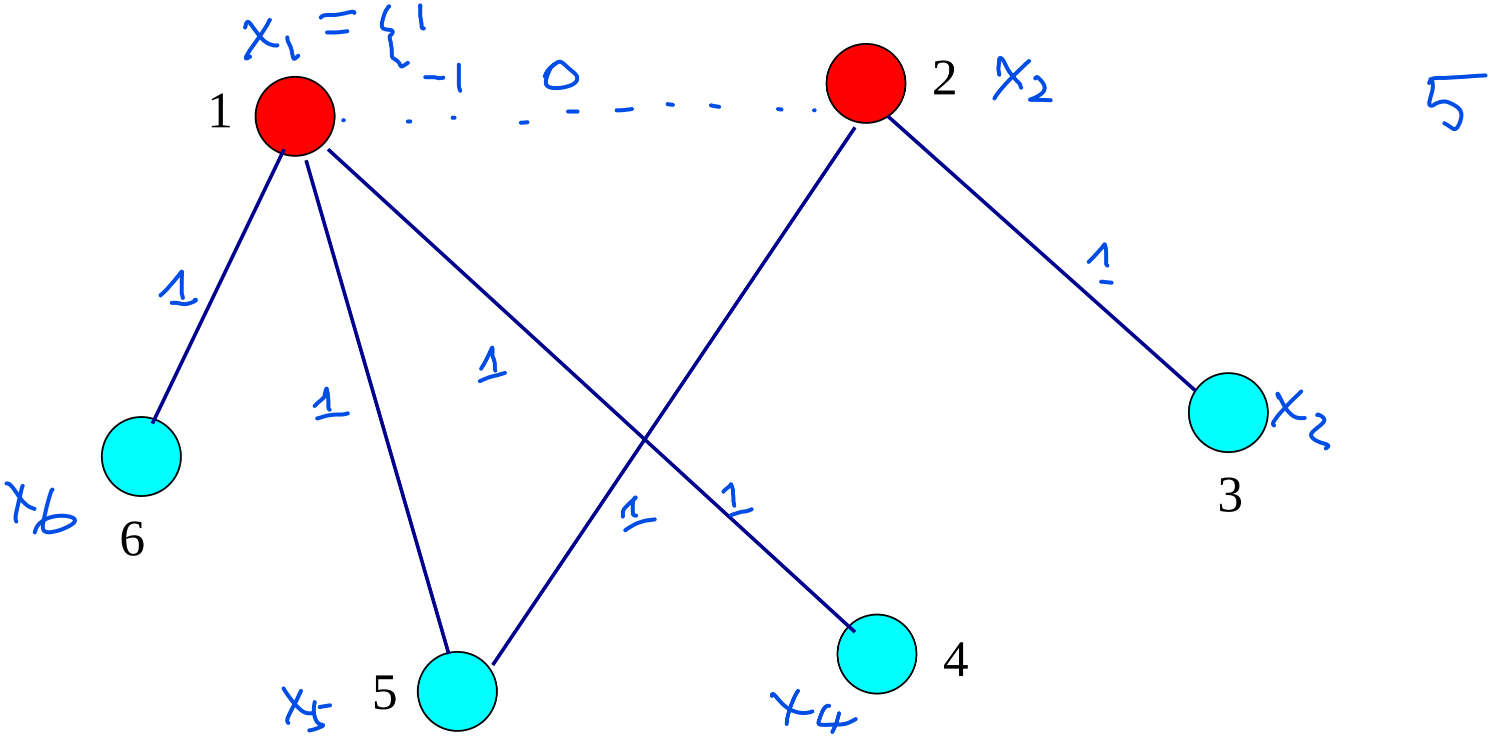


Figure 1: Illustration of the Max-Cut Problem

Max-Cut Formulation

$$w^* := \quad \text{Maximize} \quad w(\mathbf{x}) := \frac{1}{4}\sum_{i,j} w_{ij}(1 - x_i x_j)$$

(MC)

$$\text{Subject to} \quad (x_j)^2 = 1, \; j = 1, \ldots, n.$$

Semidefinite Relaxation for (MC)

$$z^{SDP} := \quad \text{Maximize} \quad \frac{1}{4}\sum_{i,j} w_{ij}(1 - X_{ij})$$
$$\text{Subject to} \quad X_{ii} = 1, \quad i = 1, \ldots, n,$$
$$X \succeq \mathbf{0}.$$

When X constrained to be rank-one or $X = \mathbf{x}\mathbf{x}^T$, the SDP formulation is equivalent to the original problem.

Let $\bar{X}$ be an optimal solution for (SDP). Then, generate a random vector $\mathbf{u} \in N(0, \bar{X})$:

$$\hat{\mathbf{x}} = \text{Sign}(\mathbf{u}), \qquad \mathsf{E}[\hat{x}_i \hat{x}_j] = \arcsin(\bar{X}_{ij})$$

Theorem 1 *(Goemans and Williamson)*

$$\mathsf{E}[w(\hat{\mathbf{x}})] \geq .878 z^{SDP} \geq .878 w^*.$$

Max-Bisection Formulation

$$w^* := \quad \text{Maximize} \quad w(\mathbf{x}) := \frac{1}{4}\sum_{i,j} w_{ij}(1 - x_i x_j)$$

(MB)

$$\text{Subject to} \quad (x_i)^2 = 1,\ i = 1, \ldots, n,$$

$$\sum_{i=1}^{n} x_i = 0.$$

What complicates matters in Max-Bisection, comparing to Max-Cut, is that two objectives are actually sought—the objective value of $w(\mathbf{x})$ and the size balance $\sum_i x_i$. Therefore, in any (randomized) rounding method at the beginning, we need to balance the (expected) quality of $w(\hat{\mathbf{x}})$ and the (expected) size balance of $\sum_i \hat{x}_i$.

Semidefinite Relaxation for (MB)

$$z^{SDP} := \quad \text{Maximize} \quad \frac{1}{4}\sum_{i,j} w_{ij}(1 - X_{ij})$$
$$\text{Subject to} \quad X_{ii} = 1, \quad i = 1, \ldots, n,$$
$$\textstyle\sum_{ij} X_{ij} = 0,$$
$$X \succeq \mathbf{0}.$$

Theorem 2 *(Y 1994) There is a randomized algorithm that generates a bisection solution* $\hat{\mathbf{x}}$ *from the SDP relaxation such that*

$$\mathrm{E}[w(\hat{\mathbf{x}})] \geq .699 z^{SDP} \geq .699 w^{*}.$$

Graph Realization and Sensor Network Localization

Given a graph $G = (V, E)$ and sets of non–negative weights, say $\{d_{ij} : (i, j) \in E\}$, the goal is to compute a realization of G in the Euclidean space $\mathbf{R}^d$ for a given low dimension d, where the distance information is preserved.

More precisely: given anchors $\mathbf{a}_k \in \mathbf{R}^d$, $d_{ij} \in N_x$, and $\hat{d}_{kj} \in N_a$, find $\mathbf{x}_i \in \mathbf{R}^d$ such that

$$
\begin{aligned}
\|\mathbf{x}_i - \mathbf{x}_j\|^2 &= d_{ij}^2, \ \forall\,(i, j) \in N_x,\ i < j, \\
\|\mathbf{a}_k - \mathbf{x}_j\|^2 &= \hat{d}_{kj}^2, \ \forall\,(k, j) \in N_a.
\end{aligned}
$$

This is a set of Quadratic Equations, which can be represented as an optimization problem:

$$
\min_{\mathbf{x}_i \forall i} \sum_{(i,j)\in N_x} (\|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2)^2 + \sum_{(k,j)\in N_a} (\|\mathbf{a}_k - \mathbf{x}_j\|^2 - \hat{d}_{kj}^2)^2.
$$

Does the system have a localization or realization of all $\mathbf{x}_j$'s? Is the localization unique? Is there a certification for the solution to make it reliable or trustworthy? Is the system partially localizable with a certification?

It can be relaxed to SOCP (change "$=$" to "$\leq$") or SDP.

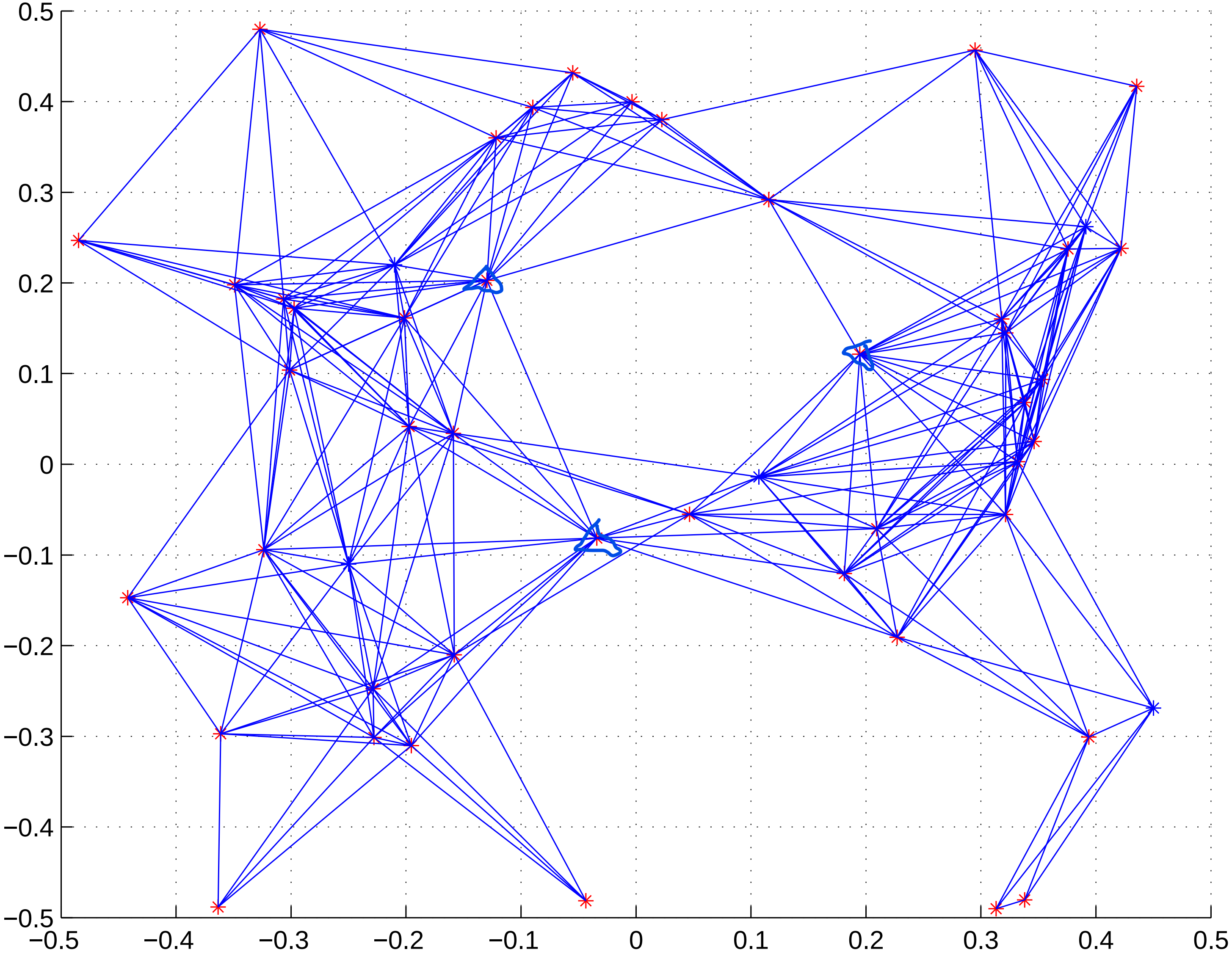


Figure 2: 50-node 2-D Sensor Localization.

Matrix Representation of SNL and SDP Relaxation

Let $X = [\mathbf{x}_1\ \mathbf{x}_2\ ...\ \mathbf{x}_n]$ be the $d \times n$ matrix that needs to be determined and $\mathbf{e}_j$ be the vector of all zero except 1 at the jth position. Then

$$\mathbf{x}_i - \mathbf{x}_j = X(\mathbf{e}_i - \mathbf{e}_j) \quad \text{and} \quad \mathbf{a}_k - \mathbf{x}_j = [I\ \ X](\mathbf{a}_k; -\mathbf{e}_j)$$

so that

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{e}_i - \mathbf{e}_j)^T X^T X (\mathbf{e}_i - \mathbf{e}_j)$$

$$\|\mathbf{a}_k - \mathbf{x}_j\|^2 = (\mathbf{a}_k; -\mathbf{e}_j)^T [I\ \ X]^T [I\ \ X](\mathbf{a}_k; -\mathbf{e}_j) =$$

$$(\mathbf{a}_k; -\mathbf{e}_j)^T \begin{pmatrix} I & X \\ X^T & X^T X \end{pmatrix} (\mathbf{a}_k; -\mathbf{e}_j).$$

Or, equivalently,

$$
\begin{aligned}
&(\mathbf{e}_i - \mathbf{e}_j)^T Y (\mathbf{e}_i - \mathbf{e}_j) = d_{ij}^2, \ \forall\, i, j \in N_x, \ i < j, \\
&(\mathbf{a}_k; -\mathbf{e}_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (\mathbf{a}_k; -\mathbf{e}_j) = \hat{d}_{kj}^2, \ \forall\, k, j \in N_a, \\
&Y = X^T X.
\end{aligned}
$$

Relax $Y = X^T X$ to $Y \succeq X^T X$, which is equivalent to matrix inequality:

$$
\begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succeq \mathbf{0}.
$$

This matrix has rank at least d; if it's d, then $Y = X^T X$, and the converse is also true.

The problem is now an SDP problem: when the SDP relaxation is exact?

Algorithm: Convex relaxation first and steepest-descent-search second strategy?

Reinforcement Learning: Markov Decision/Game Process

- RL/MDPs provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
- Markov Game Processes (MGPs) provide a mathematical framework for modeling sequential decision-making of two-person turn-based zero-sum game.
- MDGPs are useful for studying a wide range of optimization/game problems solved via dynamic programming, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).
- Modern applications include dynamic planning under uncertainty, reinforcement learning, social networking, and almost all other stochastic dynamic/sequential decision/game problems in Mathematical, Physical, Management and Social Sciences.

MDP Stationary Policy and Cost-to-Go Value

- An MDP problem is defined by a given number of states, indexed by i, where each state has a set of actions, denoted by $\mathcal{A}_i$, to take. Each action, say $j \in \mathcal{A}_i$, is associated with an (immediate) cost c_j of taking, and a probability distribution $\mathbf{p}_j$ to transfer to all possible states at the next time period.

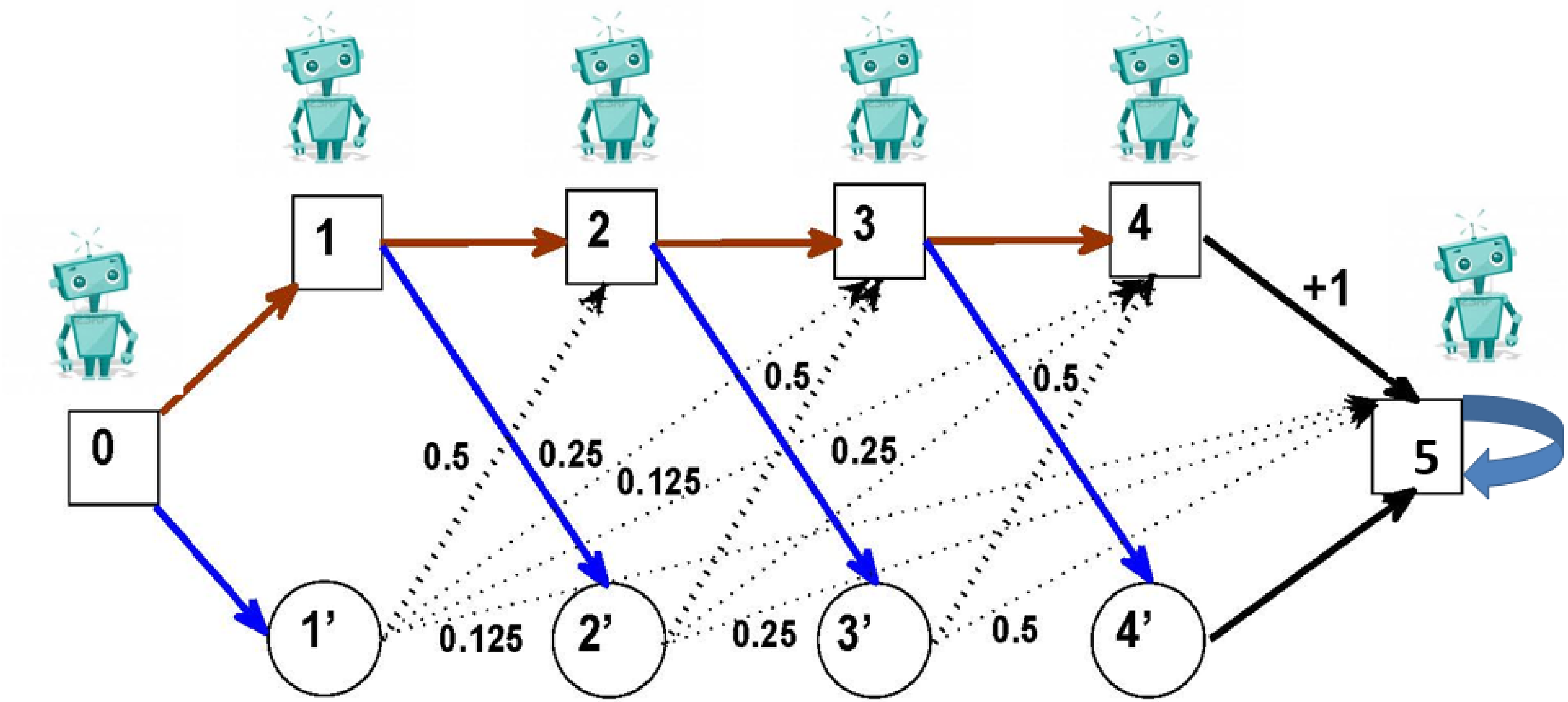
- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will take at any time period; which also lead to an expected cost-to-go value for each state: the total expected cost over all time periods if the process starts from state i and follows the policy.

- The MDP is to find a stationary policy to minimize/maximize the expected (discounted) sum over the infinite horizon with a discount factor $0 \leq \gamma < 1$:

$$\sum_{t=0}^{\infty} \gamma^t E[c^{\pi_{i^t}}(i^t, i^{t+1})].$$

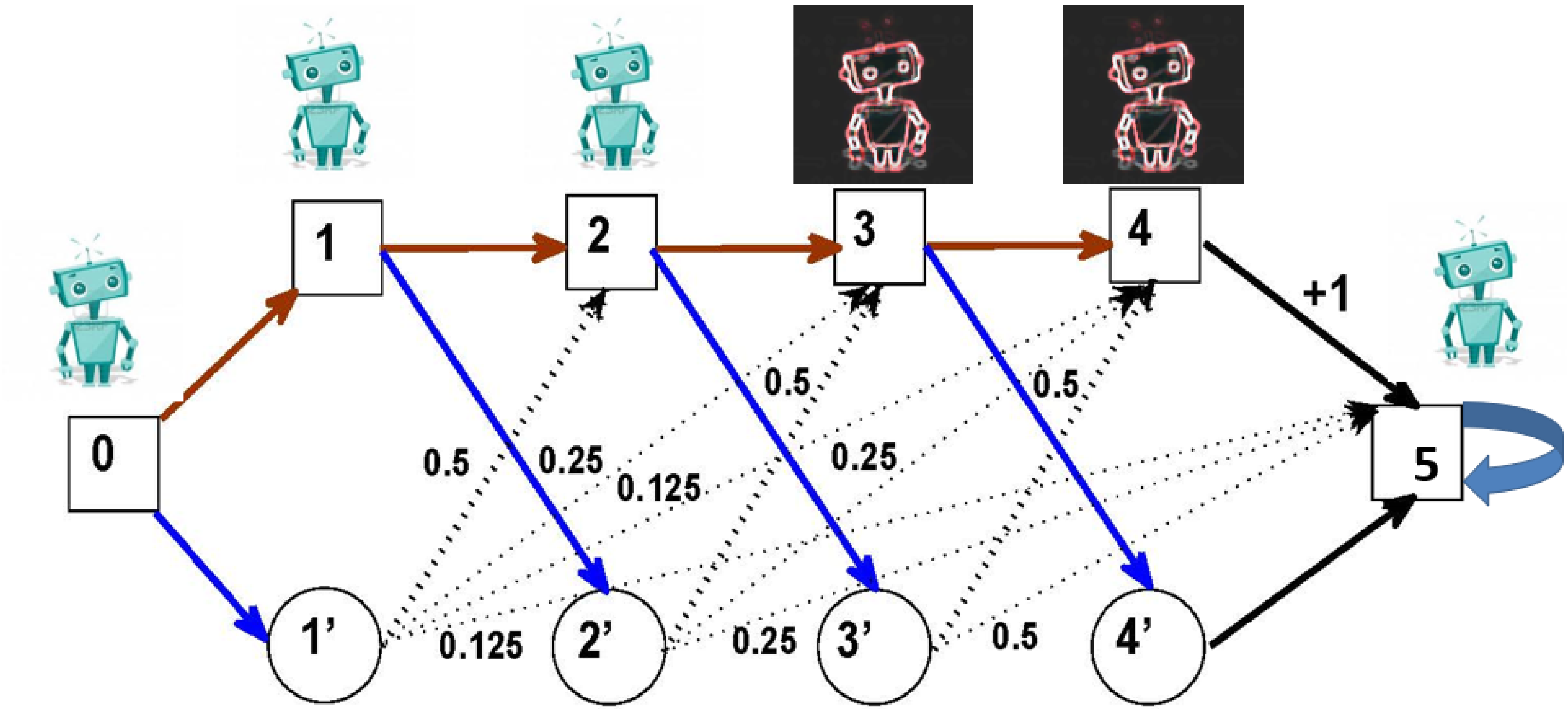
- If the states are partitioned into two sets, one is to minimize and the other is to maximize the discounted sum, then the process becomes a two-person turn-based zero-sum stochastic game.

An MDGP Toy Example: Maze Robot Runners (Simplified)



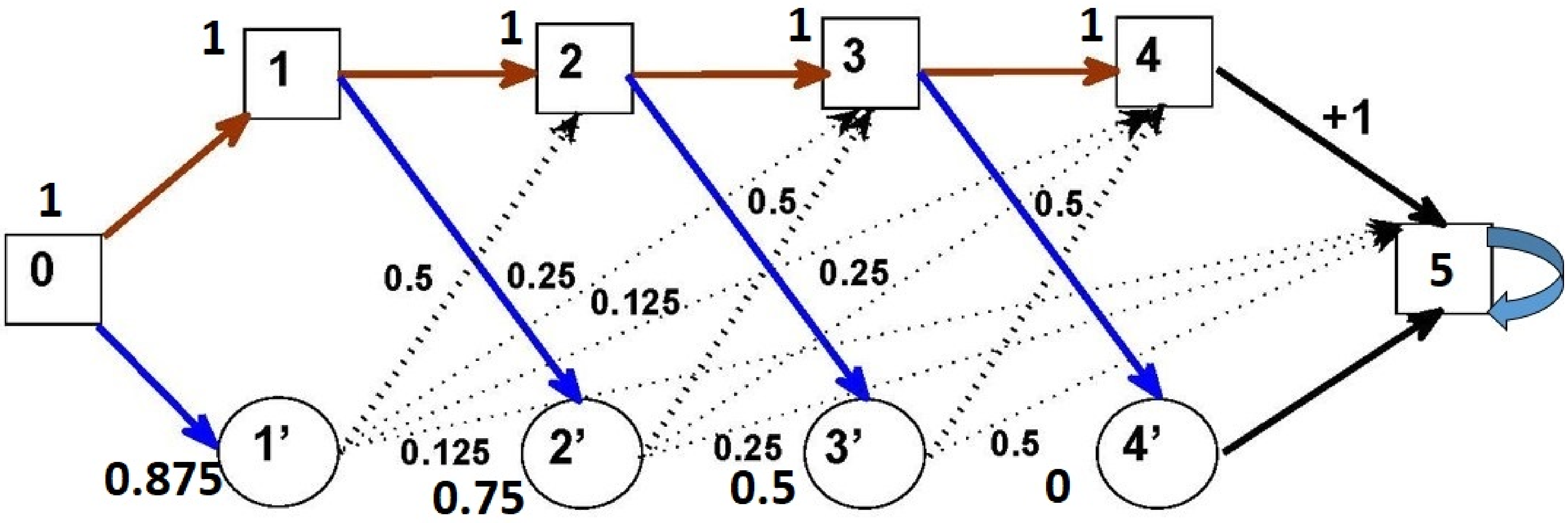
Actions are in red, blue and black; and all actions have zero cost except the state 4 to the exit/termination state 5. Which actions to take from every state to minimize the total cost (called optimal policy)?

Toy Example: Game Setting



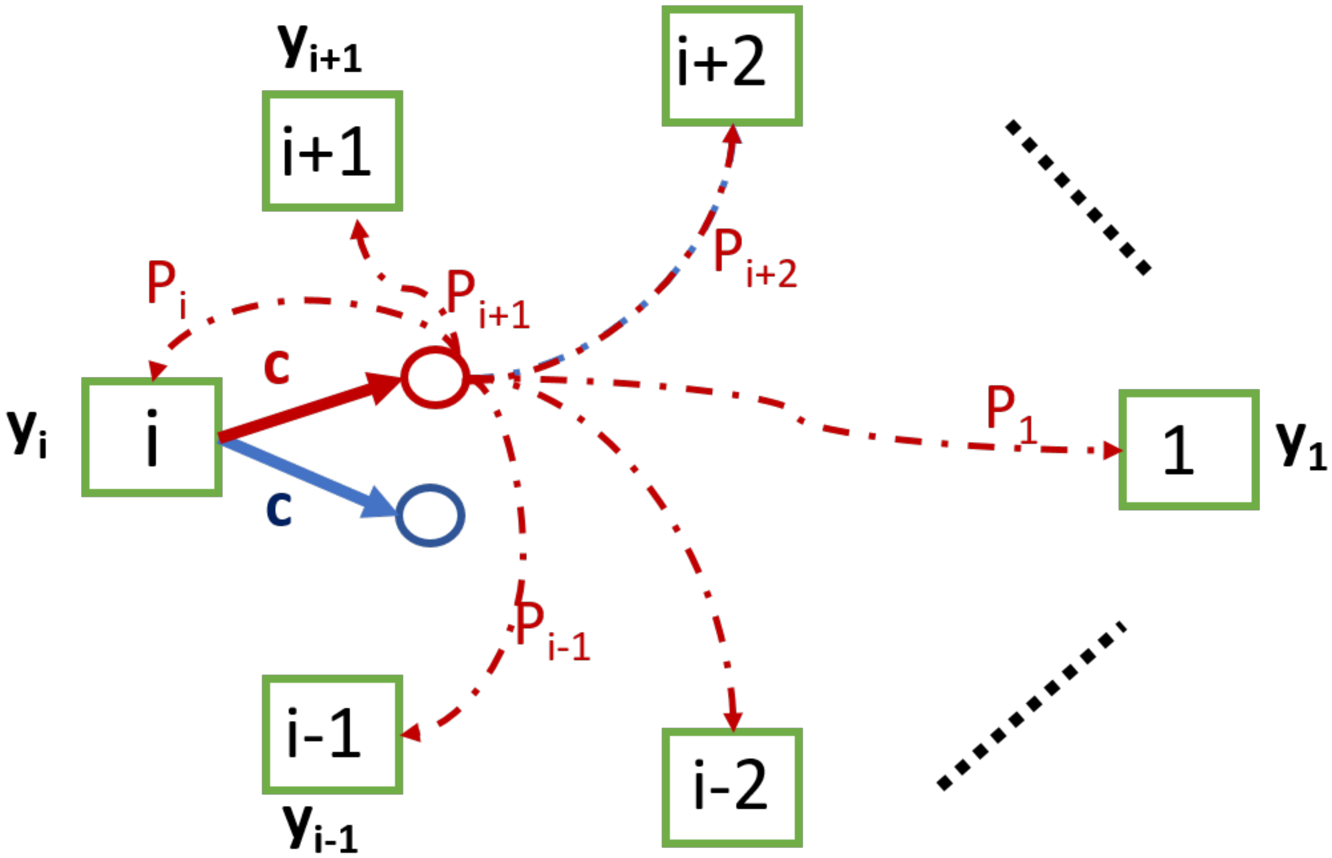
States $\{0, 1, 2, 5\}$ minimize, while States $\{3, 4\}$ maximize.

The Cost-to-Go Values of the States



Cost-to-go values on each state when actions in red are taken: the current policy is not optimal since there are better actions to choose to minimize the cost.

The Cost-to-Go Value in General



$y_i = c_j + \mathbf{p}_j^T \mathbf{y}$; when $j \in \mathcal{A}_i$ action is taken.

The Optimal Cost-to-Go Value Vector

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, ith entry for ith state, of a given policy.

The MDP problem entails choosing an optimal policy where the corresponding cost-to-go value vector $\mathbf{y}^*$ satisfying:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i,$$

with optimal policy

$$\pi_i^* = \arg\min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i.$$

In the Game setting, the conditions becomes:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^-,$$

and

$$y_i^* = \max\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^+.$$

They both are fix-point or saddle-point optimization problems. The MDP problem can be cast as a linear program; see next page.

The Maze Runner Example

The Fixed-Point formulation:

$$
\begin{aligned}
y_0 &= \min\{0+\gamma y_1, 0+\gamma(0.5y_2+0.25y_3+0.125y_4+0.125y_5)\}\\
y_1 &= \min\{0+\gamma y_2, 0+\gamma(0.5y_3+0.25y_4+0.25y_5)\}\\
y_2 &= \min\{0+\gamma y_3, 0+\gamma(0.5y_4+0.5y_5)\}\\
y_3 &= \min\{0+\gamma y_4, 0+\gamma y_5\}\\
y_4 &= 1+\gamma y_5\\
y_5 &= 0\ (\text{or } y_5 = 0+\gamma y_5)
\end{aligned}
$$

The LP formulation:

$$
\begin{array}{ll}
\text{maximize}_{\mathbf{y}} & y_0+y_1+y_2+y_3+y_4+y_5 \\
\text{subject to} & \text{change each equality above into inequality}
\end{array}
$$

The Equivalent LP Formulation for MDP

In general, the fixed-point model can be reformulated as an LP:

$$\begin{array}{lrcl} \text{maximize}_{\mathbf{y}} & \sum_{i=1}^{m} y_i & & \\ \text{subject to} & y_1 - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_1 \\ & & \vdots & \\ & y_i - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_i \\ & & \vdots & \\ & y_m - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_m. \end{array}$$

Theorem 3 *When $\mathbf{y}$ is maximized, there must be at least one inequality constraint in $\mathcal{A}_i$ that becomes equal for every state i, that is, maximal $\mathbf{y}$ is a fixed point solution.*

The Interpretations of the LP Formulation

The LP variables $\mathbf{y} \in \mathbf{R}^m$ represent the expected present cost-to-go values of the m states, respectively, for a given policy.

The LP problem entails choosing variables in $\mathbf{y}$, one for each state i, that maximize $\mathbf{e}^T\mathbf{y}$ so that it is the fixed point

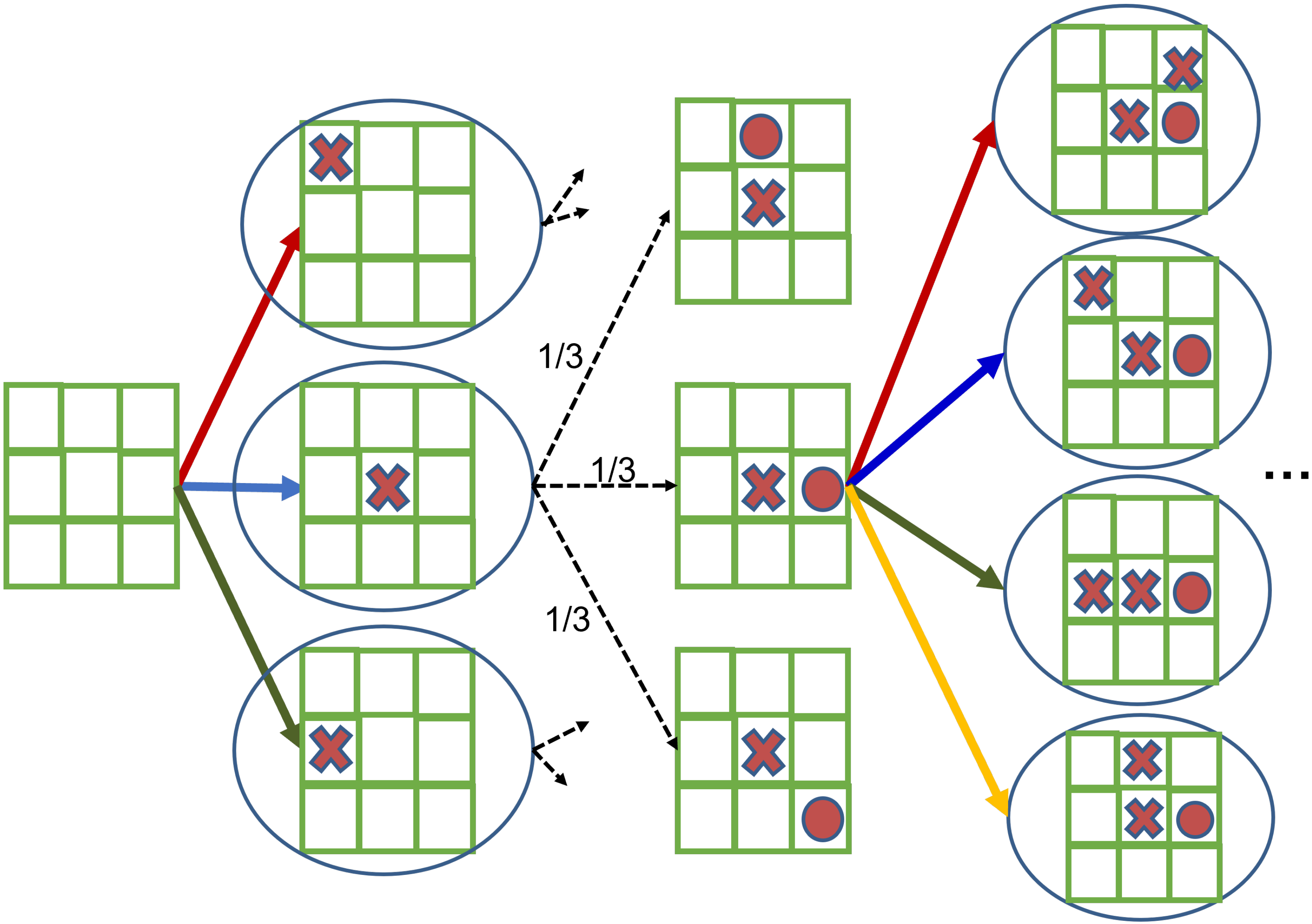
$$y_i^* = \min_{j \in \mathcal{A}_i} \{\mathbf{c}_{j_i} + \gamma \mathbf{p}_{j_i}^T \mathbf{y}\}, \ \forall i,$$

with an optimal policy

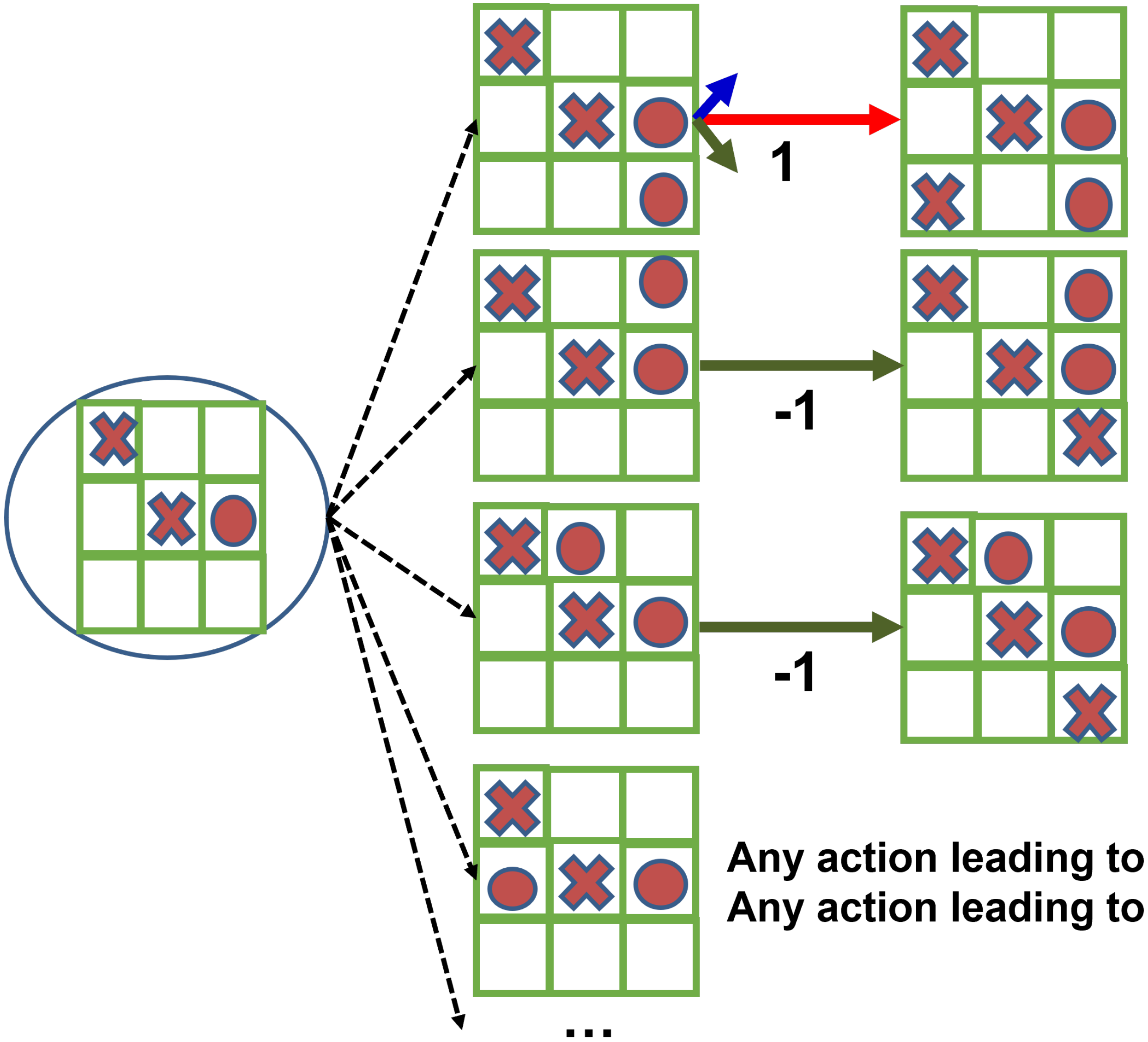
$$\pi_i^* = \arg\min\{\mathbf{c}_j + \gamma \mathbf{p}_j^T \mathbf{y}, \ j \in \mathcal{A}_i\}, \ \forall i.$$

It is well known that there exist a unique optimal stationary policy value vector $\mathbf{y}^*$ where, for each state i, y_i^* is the minimum expected present cost that an individual in state i and its progeny can incur.

States/Actions in the Tic-Tac-Toe Game



Action Costs in the Tic-Tac-Toe Game



Any action leading to win has cost -1
Any action leading to lose has cost 1